Математическое моделирование

Лабораторная работа № 2

Джафар Идрисов

Содержание

1 Цель работы **6**

2 Задание **7**

3 Выполнение лабораторной работы **8**

3.1 Нахождение расстояния x до перехода на «обход полюса" 8
3.2 Разложение скорости катера на компоненты 9
3.3 Переход к уравнению траектории 10

4 Условие задачи **11**

5 Полярные траектории: катер (ODE) и лодка (аналитически) **12**

5.1 Инициализация проекта и загрузка пакетов 12
5.2 Параметры задачи 13
5.3 Определение модели 13
5.4 Запуск 1: r0 = s/(n+1), t □ (1e-9, 8) 15
5.5 Запуск 2: r0 = s/(n-1), t □ (1e-9, 15) 15

6 Параметрическое исследование: катер (ODE) и лодка (аналитически) в полярных координатах **17**

6.1 Активация проекта и загрузка пакетов 17
6.2 Определение модели 18
6.3 Определение параметров в Dict 18
6.4 Функция-обертка для запуска одного эксперимента 19
6.5 Запуск базового эксперимента (кэширование) 21
6.6 Визуализация базового эксперимента 22
6.7 Второй базовый эксперимент (как у тебя во 2-й части): case=:minus, tmax=15 23
6.8 Параметрическое сканирование 24
6.9 Запуск всех экспериментов и сбор результатов 25
6.10 Анализ и визуализация результатов сканирования 27
6.11 Бенчмаркинг с разными параметрами 29
6.12 Сохранение всех результатов 31

7 Анализ результатов моделирования **33**

7.1 Базовые эксперименты 34
7.2 Параметрическое сканирование по n 36

7.3 Анализ метрики scale_ratio 37
7.4 Время вычислений 38

8 Выводы **39**

Список литературы **40**

Список иллюстраций

7.1 Базовый эксперимент (case=plus) 34
7.2 Базовый эксперимент (case=minus) 35
7.3 Сканирование траекторий катера 36
7.4 Зависимость scale_ratio от n 37
7.5 Зависимость времени вычисления от n 38

Список таблиц

1 Цель работы

Рассмотрим типичный пример применения математического моделирования для выбора рациональной стратегии поиска/перехвата.
Пусть в условиях тумана катер береговой охраны преследует лодку браконьеров. В некоторый момент туман рассеивается, и лодка фиксируется на расстоянии k км от катера. Далее лодка снова скрывается и продолжает движение прямолинейно в неизвестном направлении. Известно, что скорость катера в n раз выше скорости лодки.
Требуется определить траекторию движения катера, обеспечивающую перехват.

2 Задание

1. Выполнить рассуждения и получить дифференциальные уравнения движения при условии, что скорость катера в n раз больше скорости лодки.
2. Построить траектории катера и лодки для двух вариантов начальных условий.
3. По графику определить точку пересечения траекторий (момент встречи).

3 Выполнение лабораторной работы

Положим $t_0 = 0$. Точку обнаружения лодки примем за начало отсчёта. Будем считать, что лодка в момент обнаружения находится в точке $X = 0$, а катер — на расстоянии k от неё вдоль некоторого направления.

Введём полярную систему координат. Полюс — точка обнаружения лодки: $r = 0$. Полярная ось r направлена через текущее положение катера в момент наблюдения.

3.1 Нахождение расстояния x до перехода на «обход полюса»

Пусть через время t и лодка, и катер окажутся на одинаковом расстоянии x от полюса (то есть имеют одинаковый радиус $r = x$).
За это время лодка проходит путь x, а катер — $x - k$ или $x + k$ (в зависимости от взаимного расположения относительно полюса).

Обозначим скорость лодки как v, тогда скорость катера равна nv.

Для равенства времен движения получаем два случая:

- **Случай 1 (case = plus):**

$$\frac{x}{v} = \frac{x + k}{nv}.$$

- **Случай 2 (case = minus):**

$$\frac{x}{v} = \frac{x-k}{nv}.$$

Отсюда получаем значения радиуса, при которых катер «выравнивается» по расстоянию до полюса с лодкой:

$$x_1 = \frac{k}{n+1}, \quad \theta_0 = 0,$$

$$x_2 = \frac{k}{n-1}, \quad \theta_0 = -\pi.$$

Далее катер должен двигаться так, чтобы не уступать лодке в радиальном удалении от полюса, одновременно «обметая» направления вокруг полюса.

3.2 Разложение скорости катера на компоненты

Скорость катера раскладываем на радиальную и тангенциальную составляющие:

- радиальная скорость:
$$v_r = \frac{dr}{dt};$$
- тангенциальная скорость:
$$v_t = r\frac{d\theta}{dt}.$$

Чтобы катер сохранял одинаковую с лодкой скорость удаления от полюса, задаём:

$$\frac{dr}{dt} = v.$$

Полная скорость катера равна nv, следовательно по теореме Пифагора:

$$(nv)^2 = v_r^2 + v_t^2.$$

Так как $v_r = v$, то:

$$v_t = \sqrt{(nv)^2 - v^2} = v\sqrt{n^2 - 1}.$$

Отсюда:

$$r\frac{d\theta}{dt} = v\sqrt{n^2 - 1}.$$

Таким образом, исходная динамика сводится к системе ОДУ:

$$\begin{cases} \frac{dr}{dt} = v, \\ r\frac{d\theta}{dt} = v\sqrt{n^2 - 1}. \end{cases}$$

3.2.1 Начальные условия

Случай (case = plus):

$$\begin{cases} \theta_0 = 0, \\ r_0 = \frac{k}{n+1}. \end{cases}$$

Случай (case = minus):

$$\begin{cases} \theta_0 = -\pi, \\ r_0 = \frac{k}{n-1}. \end{cases}$$

3.3 Переход к уравнению траектории

Исключим t из системы. Делим первое уравнение на второе:

$$\frac{dr/dt}{d\theta/dt} = \frac{dr}{d\theta} \quad \Rightarrow \quad \frac{dr}{d\theta} = \frac{r}{\sqrt{n^2 - 1}}.$$

Начальные условия сохраняются (по выбранному случаю). Решение этого уравнения даёт траекторию катера в полярных координатах.

После получения аналитической формы (и/или численного решения) строим траектории катера и лодки для двух наборов начальных данных и определяем точку пересечения.

4 Условие задачи

В тумане катер береговой охраны преследует лодку браконьеров. В момент видимости лодка обнаружена на расстоянии $k = 20$ км от катера. Затем лодка уходит прямолинейно в неизвестном направлении.
Известно, что скорость катера в $n = 5$ раз больше скорости лодки.

Для моделирования процесса и построения графиков использовались внешние файлы с программным кодом:

5 Полярные траектории: катер (ODE) и лодка (аналитически)

Цель: построить траектории в полярных координатах для «катера» (решение ОДУ) и «лодки» (заданная прямая в декартовых координатах, переведённая в полярные).

5.1 Инициализация проекта и загрузка пакетов

```
using DrWatson
@quickactivate "project"

using DifferentialEquations
using Plots
using DataFrames
using JLD2

script_name = isempty(PROGRAM_FILE) ? "interactive" : splitext(basename(PROGRAM_FILE)
mkpath(plotsdir(script_name))
mkpath(datadir(script_name))
```

5.2 Параметры задачи

```
n  = 5
s  = 20
fi = 3/4 * pi

p = (n = n, s = s, fi = fi)
```

5.3 Определение модели

Катер: dr/dθ = r / sqrt(n^2 - 1) Здесь независимая переменная — θ (мы используем t как θ, как принято в ODEProblem).

```
function cutter_ode!(dr, r, p, □)
    dr[1] = r[1] / sqrt(p.n^2 - 1)
end
```

Лодка: линия x(t)=t, y(t)=tan(fi+pi)*t, затем перевод в (r, θ) В Julia правильнее использовать atan(y, x), чтобы угол был в верном квадранте.

```
function boat_polar(t, p)
    k = tan(p.fi + pi)
    x = t
    y = k * t
    r = hypot(x, y)          # sqrt(x^2 + y^2)
    □ = atan(y, x)
    return r, □
end
```

Утилита: генерация траектории лодки для набора t

```julia
function make_boat_curve(tgrid, p)
    r = VectorFloat64(undef, length(tgrid))
    □ = VectorFloat64(undef, length(tgrid))
    for (i, tt) in pairs(tgrid)
        ri, □i = boat_polar(tt, p)
        r[i] = ri
        □[i] = □i
    end
    return r, □
end
```

Утилита: один прогон (и график, и таблицы, и сохранение)

```julia
function run_case(case_name; r0, □span=(0.0, 2pi), n□=10_000, tmin=1e-9, tmax=8.0, nt
```

— Катер (ODE) —

```julia
    □grid = collect(LinRange(□span[1], □span[2], n□))
    prob = ODEProblem(cutter_ode!, [r0], □span, p)
    sol  = solve(prob, Tsit5(), saveat=□grid)

    df_cutter = DataFrame(□ = sol.t, r = first.(sol.u))
```

— Лодка (аналитически) —

```julia
    tgrid = collect(LinRange(tmin, tmax, nt))
    r_boat, □_boat = make_boat_curve(tgrid, p)
    df_boat = DataFrame(t = tgrid, □ = □_boat, r = r_boat)
```

— Визуализация —

```
    plt = plot(sol, proj=:polar, label="катер", xlabel="", ylabel="r",
                title="Полярные траектории — $case_name", lw=2, legend=:topleft)
    plot!(plt, _boat, r_boat, proj=:polar, label="лодка", lw=2)
```

— Сохранение —

```
    savefig(plt, plotsdir(script_name, "polar_$case_name.png"))
    @save datadir(script_name, "data_$case_name.jld2") df_cutter df_boat p r0 span r

    return (plt=plt, df_cutter=df_cutter, df_boat=df_boat)
end
```

5.4 Запуск 1: r0 = s/(n+1), t (1e-9, 8)

```
case1_r0 = p.s / (p.n + 1)
res1 = run_case("r0=s_div_(n+1)"; r0=case1_r0, tmax=8.0, p=p)

println("Кейс 1 — первые 5 строк (катер):")
println(first(res1.df_cutter, 5))
println("\nКейс 1 — первые 5 строк (лодка):")
println(first(res1.df_boat, 5))
```

5.5 Запуск 2: r0 = s/(n-1), t (1e-9, 15)

```
case2_r0 = p.s / (p.n - 1)
res2 = run_case("r0=s_div_(n-1)"; r0=case2_r0, tmax=15.0, p=p)
```

```
println("\nКейс 2 — первые 5 строк (катер):")
println(first(res2.df_cutter, 5))
println("\nКейс 2 — первые 5 строк (лодка):")
println(first(res2.df_boat, 5))
```

(опционально) показать последний график в интерактивной среде

```
res2.plt
```

6 Параметрическое исследование: катер (ODE) и лодка (аналитически) в полярных координатах

6.1 Активация проекта и загрузка пакетов

```
using DrWatson
@quickactivate "project"

using DifferentialEquations
using DataFrames
using Plots
using JLD2
using BenchmarkTools
```

Установка каталогов

```
script_name = splitext(basename(PROGRAM_FILE))[1]
mkpath(plotsdir(script_name))
mkpath(datadir(script_name))
```

6.2 Определение модели

Катер: dr/dθ = r / sqrt(n^2 - 1)

```
function cutter_ode!(dr, r, p, □)
    dr[1] = r[1] / sqrt(p.n^2 - 1)
end
```

Лодка: x=t, y=tan(fi+pi)*t -> (r, θ) Важно: используем atan(y, x), чтобы угол был корректен по квадрантам

```
function boat_polar(t, p)
    k = tan(p.fi + pi)
    x = t
    y = k * t
    r = hypot(x, y)
    □ = atan(y, x)
    return r, □
end
```

6.3 Определение параметров в Dict

Все параметры — в одном Dict, как в шаблоне. Здесь «case» управляет выбором r0: :plus (s/(n+1)) или :minus (s/(n-1)).

```
base_params = Dict(
    :n => 5,
    :s => 20,
    :fi => 3/4*pi,
```

```
    :case => :plus,                   # :plus или :minus
    :□span => (0.0, 2*pi),             # интервал по □
    :n□ => 10_000,                     # число точек для сохранения решения катера

    :tspan_boat => (1e-9, 8.0),        # интервал по t для лодки
    :nt_boat => 1_000,                 # число точек для лодки

    :solver => Tsit5(),
    :experiment_name => "base_experiment"
)

println("Базовые параметры эксперимента:")
for (key, value) in base_params
    println(" $key = $value")
end
```

6.4 Функция-обертка для запуска одного эксперимента

Возвращаем Dict со строковыми ключами (как в твоем шаблоне).

```
function run_single_experiment(params::Dict)
    @unpack n, s, fi, case, □span, n□, tspan_boat, nt_boat, solver = params
```

Параметры для ODE в виде именованного кортежа

```
    p = (n=n, s=s, fi=fi)
```

Начальное условие r0 зависит от кейса

```
r0 = case == :plus ? s/(n+1) : s/(n-1)
```

— Катер (ODE) —

```
□grid = collect(LinRange(□span[1], □span[2], n□))
prob = ODEProblem(cutter_ode!, [r0], □span, p)
sol = solve(prob, solver; saveat=□grid)

r_cutter = first.(sol.u)
```

— Лодка (аналитика) —

```
tgrid = collect(LinRange(tspan_boat[1], tspan_boat[2], nt_boat))
r_boat = VectorFloat64(undef, length(tgrid))
□_boat = VectorFloat64(undef, length(tgrid))
for (i, tt) in pairs(tgrid)
    ri, □i = boat_polar(tt, p)
    r_boat[i] = ri
    □_boat[i] = □i
end
```

— Мини-анализ (несколько метрик, чтобы было что сравнивать в сканировании) — 1) финальный радиус катера на θ=end

```
r_cutter_final = r_cutter[end]
```

2) максимальный радиус лодки на её сетке

```
r_boat_max = maximum(r_boat)
```

3) простая «разница масштабов» (без строгого физического смысла, но для сравнения кейсов полезно)

```
    scale_ratio = r_cutter_final / r_boat_max

    return Dict(
        "solution" => sol,
        "□_points" => sol.t,
        "r_cutter" => r_cutter,

        "t_points_boat" => tgrid,
        "□_boat" => □_boat,
        "r_boat" => r_boat,

        "r0" => r0,
        "r_cutter_final" => r_cutter_final,
        "r_boat_max" => r_boat_max,
        "scale_ratio" => scale_ratio,

        "parameters" => params
    )
end
```

6.5 Запуск базового эксперимента (кэширование)

```
data_base, path_base = produce_or_load(
    datadir(script_name, "single"),
    base_params,
    run_single_experiment;
    prefix = "polar",
```

```
    tag = false,
    verbose = true
)

println(”\nРезультаты базового эксперимента:”)
println(” r0: ”, data_base[”r0”])
println(” r_cutter_final: ”, data_base[”r_cutter_final”])
println(” r_boat_max: ”, data_base[”r_boat_max”])
println(” scale_ratio: ”, round(data_base[”scale_ratio”]; digits=4))
println(” Файл результатов: ”, path_base)
```

6.6 Визуализация базового эксперимента

```
p1 = plot(
    data_base[”□_points”], data_base[”r_cutter”],
    proj=:polar,
    label=”катер”,
    xlabel=”□”,
    ylabel=”r”,
    title=”Базовый эксперимент (case=$(base_params[:case]))”,
    lw=2,
    legend=:topleft,
    grid=true
)
plot!(
    p1,
    data_base[”□_boat”], data_base[”r_boat”],
```

```
    proj=:polar,
    label=”лодка”,
    lw=2
)

savefig(p1, plotsdir(script_name, ”single_experiment.png”))
```

6.7 Второй базовый эксперимент (как у тебя во 2-й части): case=:minus, tmax=15

```
base_params2 = copy(base_params)
base_params2[:case] = :minus
base_params2[:tspan_boat] = (1e-9, 15.0)
base_params2[:experiment_name] = ”base_experiment_minus”

data_base2, path_base2 = produce_or_load(
    datadir(script_name, ”single”),
    base_params2,
    run_single_experiment;
    prefix = ”polar”,
    tag = false,
    verbose = true
)

p1b = plot(
    data_base2[”□_points”], data_base2[”r_cutter”],
    proj=:polar,
```

```
    label=”катер”,
    xlabel=”□”,
    ylabel=”r”,
    title=”Базовый эксперимент (case=$(base_params2[:case]))”,
    lw=2,
    legend=:topleft,
    grid=true
)
plot!(
    p1b,
    data_base2[”□_boat”], data_base2[”r_boat”],
    proj=:polar,
    label=”лодка”,
    lw=2
)
savefig(p1b, plotsdir(script_name, ”single_experiment_minus.png”))
```

6.8 Параметрическое сканирование

В твоей задаче естественно сканировать n (оно влияет и на ODE, и на r0). При желании можно заменить на :fi или :s (или сделать сетку по двум параметрам).

```
param_grid = Dict(
    :n => [3, 4, 5, 6, 8, 10],        # сканируем n
    :s => [20],                       # фиксируем
    :fi => [3/4*pi],                  # фиксируем

    :case => [:plus, :minus],         # сканируем оба кейса
```

```
    :□span => [(0.0, 2*pi)],
    :n□ => [10_000],

    :tspan_boat => [(1e-9, 8.0)],    # можно тоже сканировать, но обычно фиксируют
    :nt_boat => [1_000],

    :solver => [Tsit5()],
    :experiment_name => [”parametric_scan”]
)

all_params = dict_list(param_grid)

println(”\n” * ”=”^60)
println(”ПАРАМЕТРИЧЕСКОЕ СКАНИРОВАНИЕ”)
println(”Всего комбинаций параметров: ”, length(all_params))
println(”Исследуемые n: ”, param_grid[:n])
println(”Исследуемые case: ”, param_grid[:case])
println(”=”^60)
```

6.9 Запуск всех экспериментов и сбор результатов

```
all_results = []
all_dfs = []

for (i, params) in enumerate(all_params)
    println(”Прогресс: $i/$(length(all_params)) | n=$(params[:n]) | case=$(params[:ca
```

```
    data, path = produce_or_load(
        datadir(script_name, "parametric_scan"),
        params,
        run_single_experiment;
        prefix = "scan",
        tag = false,
        verbose = false
    )
```

Сводка по эксперименту

```
    result_summary = merge(
        params,
        Dict(
            :r0 => data["r0"],
            :r_cutter_final => data["r_cutter_final"],
            :r_boat_max => data["r_boat_max"],
            :scale_ratio => data["scale_ratio"],
            :filepath => path
        )
    )
    push!(all_results, result_summary)
```

Полные данные (катер) — удобно для дальнейших графиков/анализа

```
    df = DataFrame(
        □ = data["□_points"],
        r = data["r_cutter"],
        n = fill(params[:n], length(data["□_points"])),
        case = fill(string(params[:case]), length(data["□_points"]))
```

```
    )
    push!(all_dfs, df)
end
```

6.10 Анализ и визуализация результатов сканирования

```
results_df = DataFrame(all_results)
println(”\nСводная таблица результатов (первые строки):”)
println(first(results_df, 10))
```

Сравнительный график траекторий катера для всех комбинаций (по θ)

```
p2 = plot(size=(900, 520), dpi=150)
for params in all_params
    data, _ = produce_or_load(
        datadir(script_name, ”parametric_scan”),
        params,
        run_single_experiment;
        prefix = ”scan”,
        tag = false,
        verbose = false
    )

    plot!(
        p2,
        data[”□_points”], data[”r_cutter”],
        label=”n=$(params[:n]), case=$(params[:case])”,
```

```
        lw=2,
        alpha=0.8
    )
end
plot!(
    p2,
    xlabel="□",
    ylabel="r(□)",
    title="Сканирование: траектории катера (ODE) при разных n и case",
    legend=:outerright,
    grid=true
)
savefig(p2, plotsdir(script_name, "parametric_scan_cutter_comparison.png"))
```

График метрики scale_ratio по n (раздельно для case) (scale_ratio = r_cutter_final / r_boat_max)

```
p3 = plot(size=(900, 520), dpi=150)
for cs in unique(results_df.case)
    sub = results_df[results_df.case .== cs, :]
    plot!(
        p3,
        sub.n, sub.scale_ratio,
        seriestype=:scatter,
        label="case=$cs"
    )
end
plot!(
    p3,
```

```
    xlabel=”n”,
    ylabel=”scale_ratio”,
    title=”Зависимость scale_ratio от n (для разных case)”,
    legend=:topleft,
    grid=true
)
savefig(p3, plotsdir(script_name, ”scale_ratio_vs_n.png”))
```

6.11 Бенчмаркинг с разными параметрами

```
println(”\n” * ”=”^60)
println(”Бенчмаркинг для разных n (оба case)”)
println(”=”^60)

benchmark_results = []
```

Возьмём сетку n из param_grid и оба case (как в сканировании)

```
for n_value in param_grid[:n], case_value in param_grid[:case]
    bench_params = Dict(
        :n => n_value,
        :s => base_params[:s],
        :fi => base_params[:fi],
        :case => case_value,
        :□span => base_params[:□span],
        :n□ => base_params[:n□],
        :tspan_boat => base_params[:tspan_boat],
        :nt_boat => base_params[:nt_boat],
```

```
        :solver => base_params[:solver]
    )

    function benchmark_run()
```

Только катер (ODE) — это и есть «вычислительная часть»

```
        p = (n=bench_params[:n], s=bench_params[:s], fi=bench_params[:fi])
        r0 = bench_params[:case] == :plus ? bench_params[:s]/(bench_params[:n]+1) : b
        prob = ODEProblem(cutter_ode!, [r0], bench_params[:□span], p)
        return solve(prob, bench_params[:solver]; saveat=LinRange(bench_params[:□spar
    end

    println("\nБенчмарк для n = $n_value, case = $case_value:")
    b = @benchmark $benchmark_run() samples=80 evals=1
    tsec = median(b).time / 1e9
    println(" Медианное время: ", round(tsec; digits=4), " сек")

    push!(benchmark_results, (n=n_value, case=string(case_value), time=tsec))
end

bench_df = DataFrame(benchmark_results)
```

График времени вычисления от n (раздельно по case)

```
p4 = plot(size=(900, 520), dpi=150)
for cs in unique(bench_df.case)
    sub = bench_df[bench_df.case .== cs, :]
    plot!(
        p4,
```

```
        sub.n, sub.time,
        seriestype=:scatter,
        label="case=$cs"
    )
end
plot!(
    p4,
    xlabel="n",
    ylabel="Время вычисления, сек",
    title="Зависимость времени решения ODE от n (для разных case)",
    legend=:topleft,
    grid=true
)
savefig(p4, plotsdir(script_name, "computation_time_vs_n.png"))
```

6.12 Сохранение всех результатов

```
@save datadir(script_name, "all_results.jld2") base_params base_params2 param_grid al
@save datadir(script_name, "all_plots.jld2") p1 p1b p2 p3 p4

println("\n" * "="^60)
println("ЛАБОРАТОРНАЯ РАБОТА ЗАВЕРШЕНА")
println("="^60)
println("\nРезультаты сохранены в:")
println(" • data/$(script_name)/single/ - базовые эксперименты")
println(" • data/$(script_name)/parametric_scan/ - параметрическое сканирование")
println(" • data/$(script_name)/all_results.jld2 - сводные данные")
```

```
println(" • plots/$(script_name)/ - все графики")
println(" • data/$(script_name)/all_plots.jld2 - объекты графиков")
println("\nДля анализа результатов используйте:")
println(" using JLD2, DataFrames")
println(" @load \"data/$(script_name)/all_results.jld2\"")
println(" println(results_df)")
```

7 Анализ результатов моделирования

В рамках работы выполнено численное исследование движения катера, заданного уравнением

$$\frac{dr}{d\theta} = \frac{r}{\sqrt{n^2 - 1}},$$

а также сопоставление полученной траектории с траекторией лодки, описанной аналитически и представленной в полярных координатах.

7.1 Базовые эксперименты

7.1.1 1. Случай (case = plus)

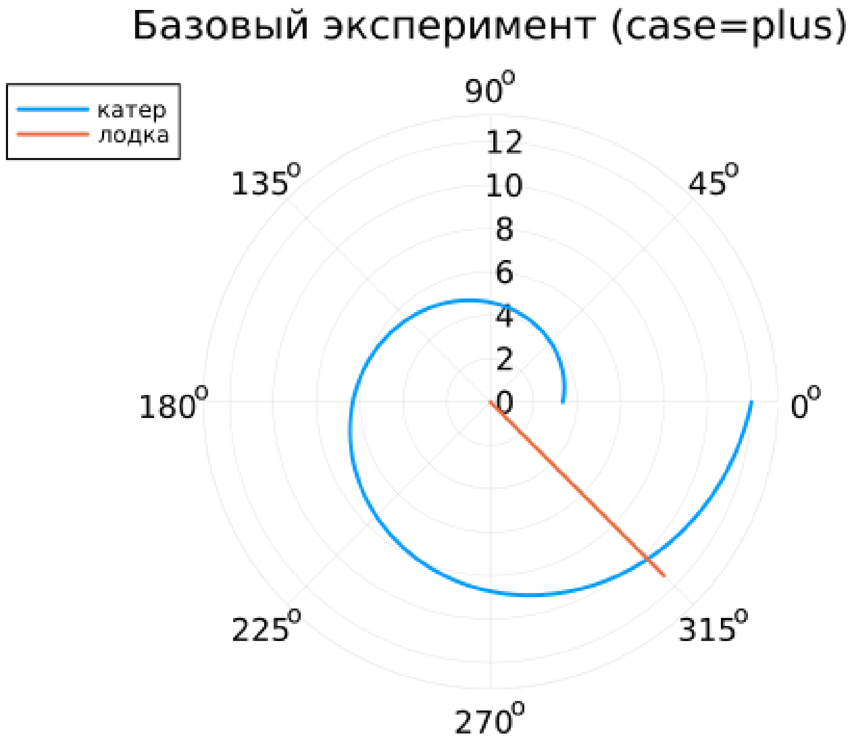

Рисунок 7.1: Базовый эксперимент (case=plus)

По полярному графику видно, что траектория катера имеет форму расходящейся спирали: при увеличении угла θ радиус r монотонно возрастает. Поскольку в правой части уравнения стоит r, рост по θ носит экспоненциальный характер.

Траектория лодки соответствует прямолинейному движению в декартовой системе; в полярных координатах это луч (линейная зависимость радиуса от параметра движения).

В рассматриваемой конфигурации катер увеличивает радиус быстрее, что визуально проявляется как расхождение кривых.

7.1.2 2. Случай (case = minus)

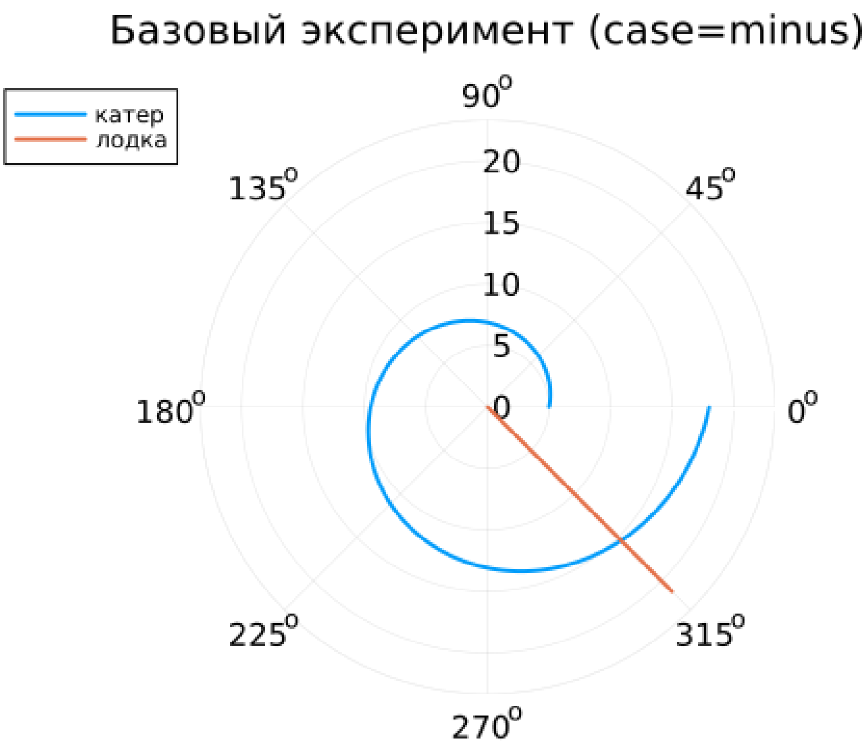

Рисунок 7.2: Базовый эксперимент (case=minus)

Во втором варианте начальный радиус больше, поэтому стартовая точка катера находится дальше от полюса.
Характер траектории сохраняется (та же «экспоненциальная» спираль), но масштаб траектории смещён наружу.

Следовательно, различие между режимами case=plus и case=minus определяется начальными условиями, а не типом роста.

7.2 Параметрическое сканирование по n

Рисунок 7.3: Сканирование траекторий катера

Исследовано влияние параметра n на динамику для обоих наборов начальных условий.

Из уравнения видно, что «интенсивность» изменения задаётся коэффициентом

$$\frac{1}{\sqrt{n^2-1}}.$$

При увеличении n этот коэффициент уменьшается, поэтому:

- при меньших n спираль расходится заметно быстрее;
- при больших n радиус растёт медленнее;
- траектория становится более «пологой» по углу.

На графике это проявляется в том, что при $n = 3$ рост наиболее быстрый, а при $n = 10$ — наиболее медленный.

7.3 Анализ метрики scale_ratio

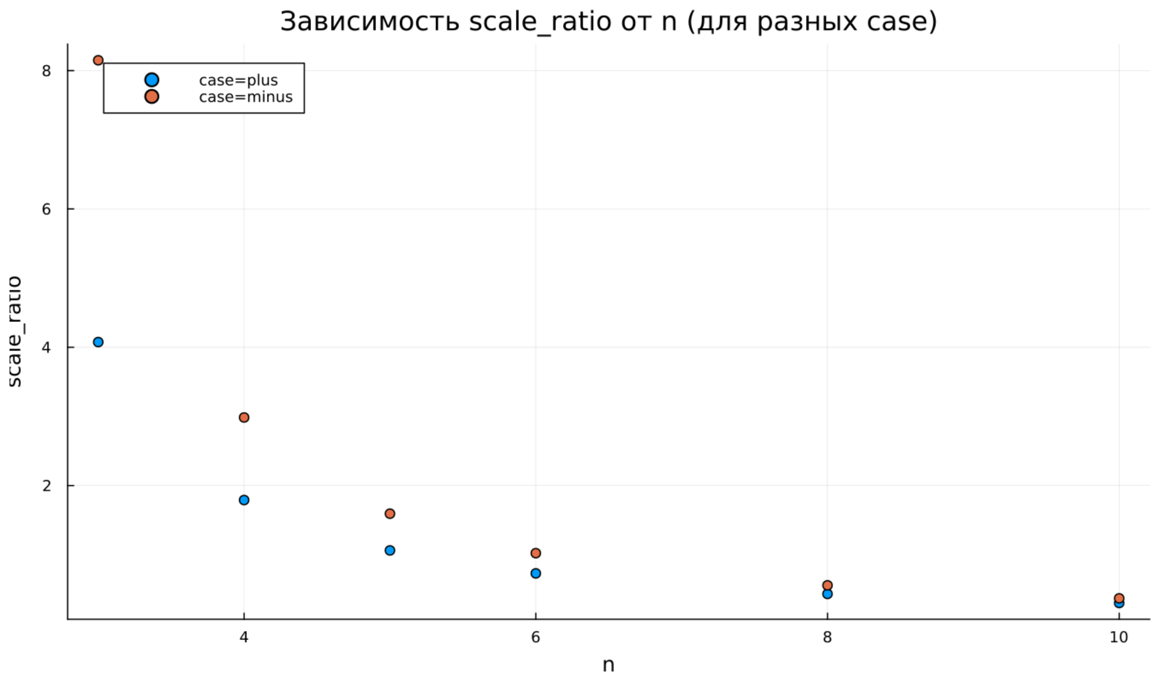

Рисунок 7.4: Зависимость scale_ratio от n

Введена метрика относительного масштаба:

$$\text{scale\_ratio} = \frac{r_{\text{final}}}{\max(r_{\text{boat}})}.$$

По графику видно:

- для малых n метрика существенно превышает 1 (катер выраженно превосходит лодку по радиальному масштабу траектории);
- с ростом n значение метрики быстро уменьшается;
- при больших n масштабы траекторий становятся сопоставимыми.

Для режима case=minus метрика выше, что объясняется увеличенным начальным радиусом.

7.4 Время вычислений

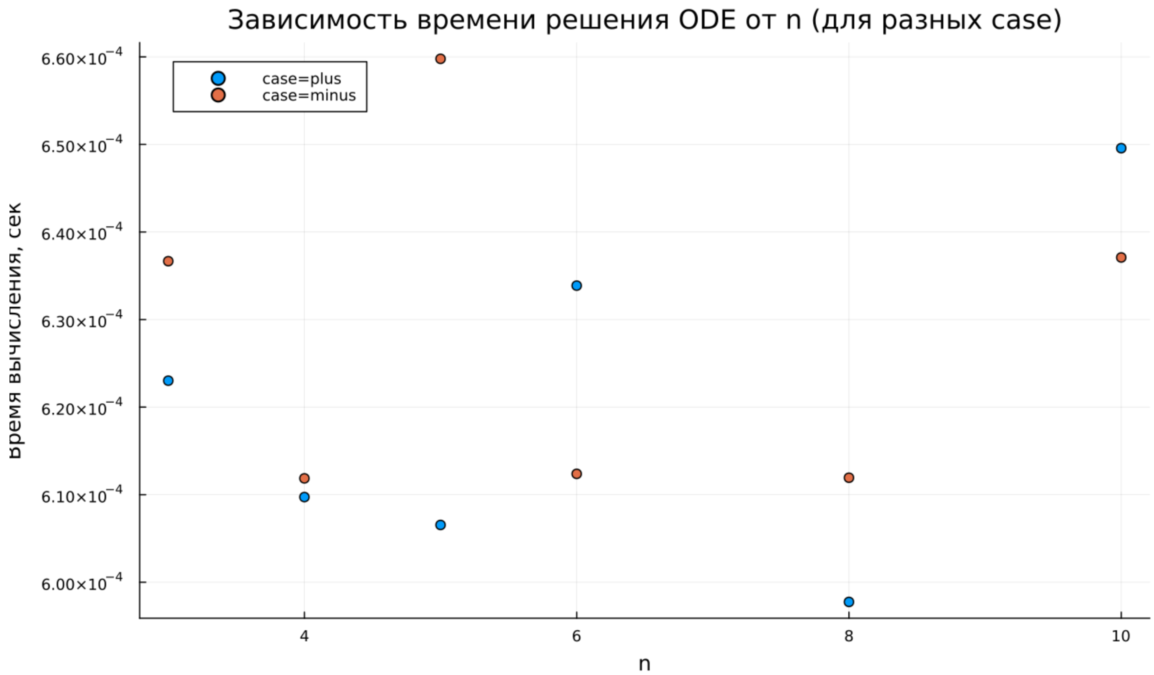

Рисунок 7.5: Зависимость времени вычисления от n

Проведён бенчмаркинг численного решения ОДУ для различных значений n.

Наблюдения:

- время расчёта порядка 6×10^{-4} секунды;
- выраженной зависимости от n не выявлено;
- небольшие колебания связаны с адаптивным выбором шага интегрирования и особенностями численной обработки.

8 Выводы

1. В полярных координатах траектория катера описывается расходящейся спиралью с экспоненциальным ростом r по углу θ.
2. Параметр n определяет темп роста: чем больше n, тем меньше коэффициент $1/\sqrt{n^2 - 1}$ и тем медленнее увеличивается радиус.
3. Выбор начальных условий (case) влияет на стартовый масштаб траектории, не меняя её качественной формы.
4. Численное решение устойчиво, а вычислительные затраты практически не зависят от параметров модели.

Полученные результаты согласуются со структурой дифференциального уравнения и ожидаемой геометрией траекторий.

Список литературы

1. Задача о погоне